www.icgst.com

Round Robin based Scheduling Algorithms, A Comparative Study

Kamal ElDahshan, Afaf Abd Elkader, Nermeen Ghazy
Dept. of mathematics, Computer science Division, Faculty of science, Al-Azhar *University*
Cairo, Egypt
[dahshan, afaf2azhar]@azhar.edu.eg, [nermeen_ghazy89]@yahoo.com
http://www.azhar.edu.eg

Abstract:
Scheduling is the process of allocating processes to the CPU in order to optimize some objective function. There are many algorithms used to schedule processes. The Round Robin (RR) CPU scheduling algorithm is one of these algorithms which is effective in time sharing and real time operating systems. It gives reasonable response time. But it suffers from several disadvantages such as high turnaround time, high waiting time and many context switches. There are large numbers of algorithms proposed to enhance the standard Round Robin algorithm. In this paper we present a survey with results analysis that conclude recommendations for an Enriched Round Robin algorithm that ameliorates the performance of average waiting time and average turnaround time.

Keywords: *Round Robin scheduling algorithm (RR), Adaptive Round Robin Scheduling algorithm, Time Quantum (TQ), Dynamic TQ, Round Robin Remaining time algorithm (RRRT), (IRR) improved Round Robin CPU Scheduling Algorithm, (AAAIRR) an improvement on the improved Round Robin CPU scheduling algorithm, (ERR) An Enhanced Round Robin CPU Scheduling Algorithm, (MMRR) Min-Max Dispersion Measure, (IRRVQ) The improved Round Robin CPU scheduling algorithm with varying time quantum, (AMRR) Average Max Round Robin Scheduling Algorithm, Average waiting time, Average turnaround time.*

Nomenclature

RR	Round Robin
TQ	Time Quantum
IRR	Improved Round Robin
AAAIRR	An improvement on the improved Round Robin
ERR	An Enhanced Round Robin CPU Scheduling
RRRT	Round Robin Remaining time algorithm
MMRR	Min-Max Dispersion Measure
IRRVQ	The improved Round Robin CPU scheduling algorithm with varying time quantum
AMRR	Average Max Round Robin
AWT	Average waiting time
ATT	Average turnaround time.
CPU	Central processing unit
ERR	Enriched Round Robin

1. Introduction
The Round Robin (RR) CPU scheduling algorithm referred as Standard RR, it is a preemptive algorithm which assigns a time slice called (time quantum TQ) [1] for each process in the ready queue. When the TQ finishes, the current process is preempted and added to the end of the ready queue [2] [3]. RR is commonly used in time sharing and real time operating system [4] [5] because it gives each process a fair share of time to use the CPU and produces low response time [6]. However, it is known that the Standard RR algorithm suffers from several disadvantages; low throughput (the number of processes that are completed per unit of time) [7], high turnaround time (the time between starting and completion) [7], high waiting time (the amount of time that is waiting in the ready queue) [7], and large number of context switches [6]. The most interesting issue with the Round Robin algorithm is the time quantum [8]. Setting a small TQ causes too many context switches which results in a low performance of the CPU [9] [10]. On the other hand, setting the TQ to a large value may cause a poor response time and the RR algorithm tends to work as FCFS [8] [9] [10].
Many algorithms improved the RR algorithm and produce better results than it. Some of these algorithms use static TQ and others use dynamic TQ. This work is a comparative study of Round Robin and its improvements.

The rest of the paper is organized as follows Section (2) focuses on some Static Time Quantum Algorithms Section (3) focuses on some Dynamic Time Quantum Algorithms Section (4) explains the Results and ameliorations Analysis Section (5) discusses the Enriched Round Robin algorithm Section (6) concludes the paper.

2. Static Time Quantum Algorithms
Some algorithms use TQ which is fixed during the execution of the algorithm. TQ may be assumed or calculated.
The following two subsections discuss both algorithms depending on assumed TQ and calculated TQ.

2.1 Description of the Round Robin algorithms with assumed Time quantum

Some researchers assumed a value for the time quantum which must not exceed the maximum burst time. The following three algorithms IRR, AAIRR, ERR [11] [12] [13] used an assumed time quantum to schedule the processes in the ready queue.

2.1.1 Improved Round Robin CPU Scheduling Algorithm (IRR)

The improved Round Robin (IRR) algorithm is similar to Round Robin (RR) with a small improvement [11]. It selects the first process that arrives in the ready queue and allocates it to the CPU for a time interval of up to 1 time quantum. After completion of process's time quantum, the remaining CPU burst time of the currently running process is checked. If the remaining CPU burst time of the currently running process is less than 1 time quantum, the CPU is again allocated to the currently running process for the remaining CPU burst time. After finishing its execution, it is removed from the ready queue and the next process will be selected. Otherwise, the process will be added to the end of the ready queue and the next process will be selected [11]. It gives better performance than RR where reduces the waiting time and turnaround time.

The pseudo code of the IRR CPU scheduling algorithm in the following:

1. START
2. Make a ready queue of the Processes say REQUEST.
3. Do 4, 5 and 6 WHILE queue REQUEST becomes empty.
4. Pick the first process from the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum.
5. If the remaining CPU burst time of the currently running process is less than 1 time quantum then allocate CPU again to the currently running process for remaining CPU burst time. After completion of execution, removed it from the ready queue and go to 3.
6. Remove the currently running process from the ready queue REQUEST and put it at the tail of the ready queue.
7. END

2.1.2 An Additional Improvement Round Robin CPU scheduling algorithm (AAIRR)

It makes an improvement on the improved Round Robin CPU scheduling algorithm. It selects the first process that arrives in the ready queue and allocates it to the CPU for a time interval of up to 1 time quantum. After completion of process's time quantum, the remaining CPU burst time of the currently running process is checked. If the remaining CPU burst time of the currently running process is less than or equal to 1 time quantum, the CPU is again allocated to the currently running process for the remaining CPU burst time and after execution it removed from the ready queue then it selects the next shortest process.

Otherwise, the process will be added to the end of the ready queue and the next process in the ready queue will be selected [12]. This algorithm reduces the waiting time and turnaround time compared to the standard Round Robin algorithm and Improved Round Robin algorithm.

The pseudo code of the AAIRR CPU scheduling algorithm in the following:

1. START
2. Make a ready queue of the Processes say REQUEST.
3. Do
4. Pick the first process that arrives to the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum.
5. If the remaining CPU burst time of the currently running process is less or equal to 1 time quantum.
6. Reallocate the CPU again to the currently running process for the remaining CPU burst time. After completing the execution, remove it from the ready queue.
7. Otherwise, remove the currently running process from the ready queue REQUEST and put it at the tail of the ready queue.
8. Pick the next shortest process from the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum then go to 5.
9. WHILE queue REQUEST in not empty
10. Calculate Average Waiting Time, Average Turnaround Time and Number of Context Switch.
11. END

2.1.3 An Enhanced Round Robin CPU Scheduling algorithm (ERR)

This algorithm makes a little improvement on the IRR algorithm. It selects the first process of the ready queue to allocate it to the CPU for a time interval of up to 1 time quantum. Then it checks the remaining burst time of the currently running process, if the remaining burst time is less than or equal to 1 time quantum, the current process is again allocated to the CPU. After completing the execution, this process is removed from the ready queue. If the remaining burst time of the currently running process is longer than 1 time quantum, the process will be added to the end of the ready queue [13].

The pseudo code of the AAIRR CPU scheduling algorithm in the following:

1. Start
2. Make a ready queue of the processes
3. Allocate the CPU to the first process of the ready queue for a time interval of up to 1 time quantum.
4. If the remaining burst time of the currently running process is less than or equal to 1 time quantum then
5. Allocate CPU again to the currently running process for the remaining burst time.


6. After completion of execution remove it from the ready queue and go to 3
7. Repeat 3,4 and 5 WHILE ready queue becomes empty
8. Remove the currently running process from the ready queue and put it at the tail of the ready queue.
9. END

1

2.1.4 Experimental data

There are three cases for incoming burst time namely; random, increasing and decreasing for randomly chosen TQ.

Case 1: The burst time of the processes is random

Consider five processes with a random burst time shown in Table 1. Assume that TQ=12ms, we apply the algorithms RR, IRR, AAIRR and ERR [11,12,13,21] to schedule these processes and the resulting Gantt charts for the RR, IRR, AAIRR and ERR algorithms are shown in Figures 1, 2, 3 and 4 respectively.

Table 1. Five processes with random burst time

process	Burst time
P1	24
P2	40
P3	5
P4	12
P5	34

a) RR algorithm

Enter the processes in the ready queue in the order; P1, P2, P3, P4 and P5. Each process is given a TQ= 12ms periodically.

Gantt chart

Figure 1: Gantt chart for RR Algorithm (Case 1)

Average waiting time= 49.2 ms
Average turnaround time= 72.2ms

b) IRR algorithm

Here the processes are scheduled according to the IRR algorithm with TQ=12.

Gantt chart

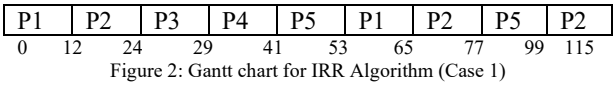

Figure 2: Gantt chart for IRR Algorithm (Case 1)

Average waiting time= 46.8 ms
Average turnaround time= 69.8 ms

c) AAIRR algorithm

Here the processes are scheduled according to the AAIRR algorithm with TQ=12

Gantt chart

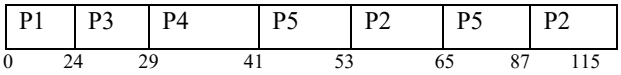

Figure 3: Gantt chart for AAIRR Algorithm (Case 1)

Average waiting time= 36.2ms
Average turnaround time= 59.2ms

d) ERR algorithm

Here the processes are scheduled according to the ERR algorithm with TQ=12

Gantt chart

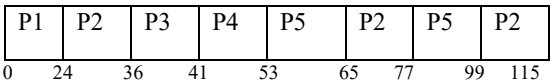

Figure 4: Gantt chart for ERR Algorithm (Case 1)

Average waiting time= 43.4 ms
Average turnaround time= 66.4 ms

The following table compares among the four algorithms RR, IRR, AAIRR and ERR with respect to average waiting time and average turnaround time.

Table 2. Comparison among the RR, IRR, AAIRR and ERR Algorithms

Algorithm	Average waiting time	Average turnaround time
RR	49.2	72.2
IRR	46.8	69.8
AAIRR	36.2	59.2
ERR	43.4	66.4

Figures 5, 6 draw the average waiting time and the average turnaround time for the four algorithms RR, IRR, AAIRR and ERR algorithms.

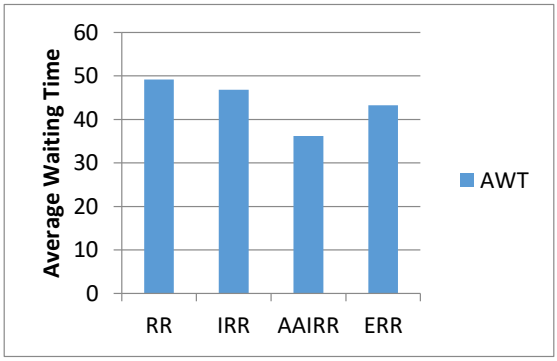

Figure 5: Graph for Average Waiting Time for RR, IRR, AAIRR and ERR algorithms (Case 1)

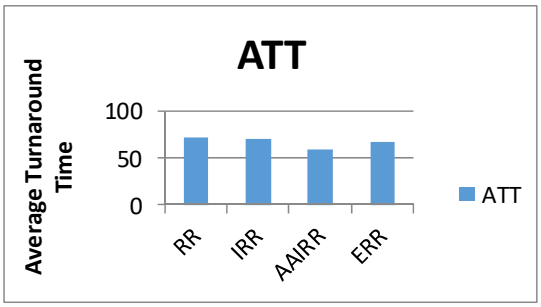

Figure 6: Graph for Average Turnaround Time for RR, IRR, AAIRR and ERR algorithms (Case 1)

From Figures 5, 6 it is noticed that the algorithms IRR, AAIRR and ERR give better results than the standard Round Robin algorithm for the average waiting time and the average turnaround time. By comparing the results of the three algorithms, it is noticed that the ERR algorithm reduces the average waiting time and average turnaround time than the IRR algorithm. However the AAIRR algorithm is the best one of them.

Case 2: the burst time of the processes is in an increasing order

Consider five processes with a randomly increasing order of burst time shown in Table 3 assume that TQ=10ms, we apply the algorithms RR, IRR, AAIRR and ERR to schedule these processes and the resulting Gantt charts for the RR, IRR, AAIRR and ERR algorithms are shown in Figures 7, 8, 9 and 10 respectively.

Table 3. Burst time in an increasing order

process	Burst time
P1	7
P2	10
P3	20
P4	26
P5	30

a) RR algorithm

Enter the processes in the ready queue in the order; P1, P2, P3, P4 and P5. Each process is given a TQ=10ms periodically.

Gantt chart

P1	P2	P3	P4	P5	P3	P4	P5	P4	P5

0 7 17 27 37 47 57 67 77 83 93

Figure 7: Gantt chart for RR Algorithm (Case 2)

Average waiting time= 32.8 ms
Average turnaround time= 51.4ms

b) IRR algorithm

Here the processes are scheduled according to the IRR algorithm with TQ=10.

Gantt chart

P1	P2	P3	P4	P5	P3	P4	P5

0 7 17 27 37 47 57 73 93

Figure 8: Gantt chart for IRR Algorithm (Case 2)

Average waiting time= 30.8 ms
Average turnaround time= 49.4 ms

c) AAIRR algorithm

Here the processes are scheduled according to the AAIRR algorithm with TQ=10.

Gantt chart

P1	P2	P3	P4	P5	P4	P5

0 7 17 37 47 57 73 93

Figure 9: Gantt chart for AAIRR Algorithm (Case 2)

Average waiting time= 26.8ms
Average turnaround time= 45.4ms

d) ERR algorithm

Here the processes are scheduled according to the ERR algorithm with TQ=10.

Gantt chart

P1	P2	P3	P4	P5	P4	P5

0 7 17 37 47 57 73 93

Figure 10: Gantt chart for ERR Algorithm (Case 2)

Average waiting time= 26.8 ms
Average turnaround time= 45.4 ms

The following table compares among the four algorithms RR, IRR, AAIRR and ERR with respect to average waiting time and average turnaround time.

Table 4. Comparison among the RR, IRR, AAIRR and ERR Algorithms

Algorithm	Average waiting time	Average turnaround time
RR	32.8	51.4
IRR	30.8	49.4
AAIRR	26.8	45.4
ERR	26.8	45.4

Figures 7, 8 draw the average waiting time and the average turnaround time for the four algorithms RR, IRR, AAIRR and ERR.

Figure 11: Graph for Average Waiting Time for RR, IRR, AAIRR and ERR algorithms (Case 2)

Figure 12: Graph for Average Turnaround Time for RR, IRR, AAIRR and ERR algorithms (Case 2)

From above Figures 11, 12 it is noticed that the algorithms IRR, AAIRR and ERR are doing better than standard Round Robin algorithm for the average waiting time and the average turnaround time. For the other three algorithms, it is noticed that the ERR algorithm gives the same results as the AAIRR algorithm because of the increasing order of the incoming processes. But the IRR algorithm gives high results than both of them. The benefit of AAIRR algorithm which is selecting the next short process to assign it to the CPU which leads to minimize the average waiting time and average turnaround time is not exist.

Case 3: The burst time of the processes is in decreasing order

Consider five processes with a randomly decreasing order of burst time shown in Table 5. Assume that TQ=15ms, we apply the algorithms RR, IRR, AAIRR and ERR to schedule these processes and the resulting Gantt charts for the RR, IRR, AAIRR and ERR algorithms are shown in Figures 13, 14, 15 and 16 respectively.

Table 5. Burst time in decreasing order

process	Burst time
P1	45
P2	36
P3	30
P4	18
P5	10

a) RR algorithm

Enter the processes in the ready queue in order; P1, P2, P3, P4 and P5. Each process is given a TQ=15ms periodically.

Gantt chart

P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2

0 15 30 45 60 70 85 100 115 118 133 139

Figure 13: Gantt chart for RR Algorithm (Case 3)

Average waiting time= 87.2 ms
Average turnaround time= 115ms

b) IRR algorithm

Here the processes are scheduled according to the IRR algorithm with TQ=15.

Gantt chart

P1	P2	P3	P4	P5	P1	P2	P3	P1

0 15 30 45 63 73 88 109 124 139

Figure 14: Gantt chart for IRR Algorithm (Case 3)

Average waiting time= 73.8 ms
Average turnaround time= 101.6 ms

c) AAIRR algorithm

Here the processes are scheduled according to the AAIRR algorithm with TQ=15.

Gantt chart

P1	P5	P4	P3	P2	P1	P2

0 15 25 43 73 88 118 139

Figure 15: Gantt chart for AAIRR Algorithm (Case 3)

Average waiting time= 51.8ms
Average turnaround time= 79.6ms

d) ERR algorithm

Here the processes are scheduled according to the ERR algorithm with TQ=15.

Gantt chart

P1	P2	P3	P4	P5	P1	P2

0 15 30 60 78 88 118 139

Figure 16: Gantt chart for ERR Algorithm (Case 3)

Average waiting time= 68.8 ms
Average turnaround time= 96.6 ms

The following table compares among the four algorithms RR, IRR, AAIRR and ERR with respect to average waiting time and average turnaround time.

Table 6. Comparison among the RR, IRR, AAIRR and ERR Algorithms

Algorithm	Average waiting time	Average Turnaround time
RR	87.2	115
IRR	73.8	101.6
AAIRR	51.8	79.6
ERR	68.8	96.6

Figures 17, 18 draw the average waiting time and the average turnaround time for the RR, IRR, AAIRR and ERR algorithms.

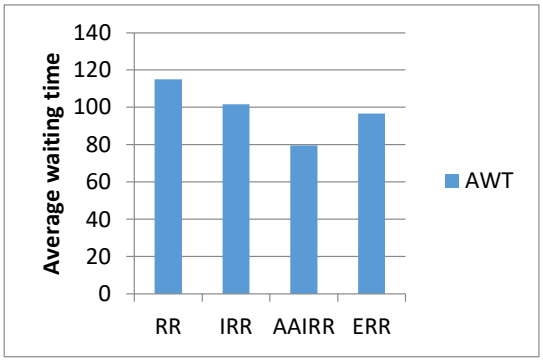

Figure 17: Graph for Average Waiting Time for RR, IRR, AAIRR and ERR algorithms (Case 3)

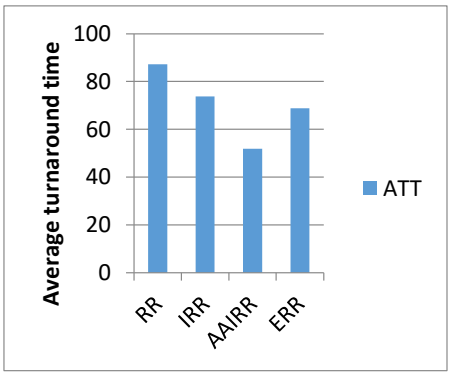

Figure 18: Graph for Average Turnaround Time for RR, IRR, AAIRR and ERR algorithms (Case 3)

From Figures 17, 18 it is noticed that the algorithms IRR, AAIRR and ERR give better results than the standard Round Robin algorithm for the average waiting time and the average turnaround time. By comparing the results of the three algorithms, it is noticed that the ERR algorithm reduces the average waiting time and average turnaround time than the IRR algorithm. However the AAIRR algorithm is the best one of them.

Hence, the AAIRR algorithm is the algorithm which gives the best results in random and decreasing cases. But in the case of increasing order it gives the same results as the ERR algorithm.

2.2 Description of the Round Robin algorithms with calculated time quantum and their Experimental data

2.2.1 Adaptive Round Robin Scheduling Algorithm (Adaptive RR)

This algorithm improved the Round Robin algorithm. It is a "priority scheduling algorithm" based on the burst time of processes. It is composed of two phases. First, the processes are arranged according to the burst time. The highest priority is given to the process with smallest burst time. This approach chooses the smart time slice depending on the number of processes. If the number of processes is even, then the smart time slice will be the average of the burst of all processes. If the number of processes is odd, then the smart time slice is equal to the burst time of the mid-process [14].

Adaptive Round Robin Pseudo code as follow.

1. First of all check ready queue is empty
2. When ready queue is empty then all the processes are assigned into the ready queue.
3. All the processes are rearranged in increasing order that means smaller burst time process get higher priority and larger burst time process get lower priority.
4. While (ready queue != NULL)
5. Calculate smart Time Slice:
   If (Number of process%2= = 0)
   STS = average CPU burst time of all processes
   Else
   STS = mid process burst time
6. Assign smart time slice to the ith process:
   Pi _ STS
7. If (i< Number of process) then go to step 6.
8. If a new process is arrived update the ready queue, go to step 2.
9. End of While
10. Calculate average waiting time, turnaround time, context switches and throughput.
11. End

2.2.2 Round Robin Remaining Time Algorithm (RRRT)

This algorithm improved the Adaptive Round Robin Scheduling algorithm. It assumes that all processes arrive at the same time in the ready queue, then they are arranged in an ascending order according to their burst time. TQ is calculated by Σ pi / 2n. If the remaining CPU burst time of the currently running process is less than the TQ, the CPU is again allocated to the currently running process for the remaining CPU burst time. Otherwise, the process will be added to the end of the ready queue [15].

The pseudo code of the Round Robin Remaining Time Algorithm:

1. Assign Process to ready queue.
2. Rearrange all the process in increasing order of their burst time
3. While (ready queue! =NULL)
4. Calculate time quantum =Σpi / 2*n
5. if (remaining burst time < time quantum)
   Allocate CPU again to the current running process for remaining burst time
   Else
   Remove the current running process from the ready queue and put it at the tail of the ready queue.
6. If no of process > 0 Go to 5
7. End while
8. Calculate average waiting time, average turnaround time.

2.1.3 Experimental data

There are three cases for incoming burst time; random, increasing and decreasing. The order of the processes affects the results of the Round Robin algorithm since the processes are executed in their given order, whereas in the other algorithms the burst time of the processes is arranged in an ascending order in all cases. The results of the three cases are described below.

Case 1: The burst time of the processes is random

Consider five processes with a random burst time shown in Table 7. We apply the algorithms RR, Adaptive RR and RRRT to schedule these processes and the resulting Gantt charts for the RR, Adaptive RR and RRRT algorithms are shown in Figures 19, 20 and 21 respectively.

Table 7. Five processes with random burst time

process	Burst time
P1	18
P2	38
P3	6
P4	28
P5	21

a) RR algorithm

Enter the processes in the ready queue in the order; P1, P2, P3, P4 and P5. Each process is given a TQ= 14ms periodically.

Gantt chart

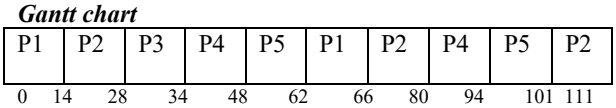

Figure 19: Gantt chart for RR Algorithm (Case 1)

Average waiting time= 59ms
Average turnaround time= 81.2ms

b) Adaptive RR algorithm

Arrange the processes in the ready queue in an ascending order of their burst time; P3=6, P1=18, P5=21, P4=28 and P2=38. Since the number of processes is odd, the time quantum is set to be the burst time of the mid process. Hence, TQ=21 according to Adaptive RR algorithm.

Gantt chart

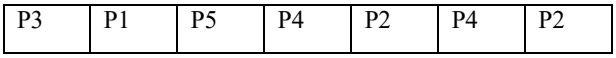

Figure 20: Gantt chart for Adaptive RR Algorithm (Case 1)

Average waiting time= 33.8 ms
Average turnaround time= 56 ms

c) RRRT algorithm

Arrange the processes in the ready queue according to their given burst time in an ascending order; P3=6, P1=18, P5=21, P4=28 and P2=38. The time quantum is calculated according to the RRRT Algorithm. The resulting TQ is 11ms.

Gantt chart

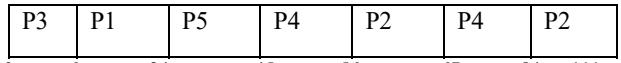

Figure 21: Gantt chart for RRRT Algorithm (Case 1)

Average waiting time= 31.8ms
Average turnaround time= 54ms

The following table compares among the three algorithms RR, Adaptive RR and RRRT with respect to average waiting time and average turnaround time.

Table 8. Comparison among the RR, Adaptive RR and RRRT Algorithms

Algorithm	Average waiting time	Average Turnaround time
RR	59	81.2
Adaptive RR	33.8	56
Round Robin Remaining Time	31.8	54

Figures 22, 23 draw the average waiting time and the average turnaround time for the RR, Adaptive RR and RRRT algorithms.

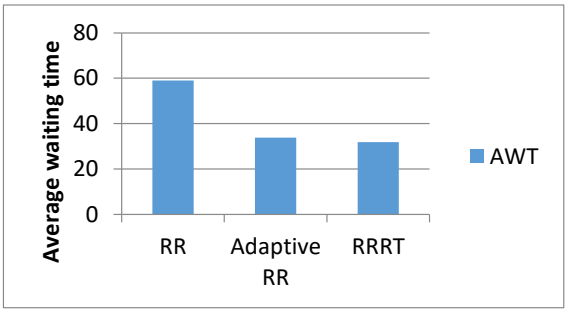

Figure 22: Graph for Average Waiting Time for RR, Adaptive RR and RRRT algorithms (Case 1)

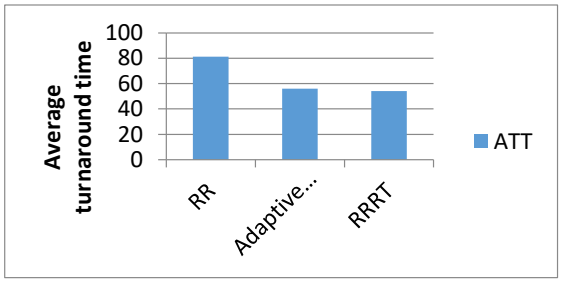

Figure 23: Graph for Average Turnaround Time for RR, Adaptive RR and RRRT algorithms (Case 1)

Case 2: The burst time of the processes is in an increasing order

Consider five processes with a randomly increasing order of burst time shown in Table 9. We apply the algorithms RR, Adaptive RR and RRRT to schedule these processes and the resulting Gantt charts for the RR, Adaptive RR and RRRT algorithms are shown in Figures 24, 25 and 26 respectively.

Table 9. Five processes with random burst time

process	Burst time
P1	15
P2	19
P3	23
P4	47
P5	56

a) RR algorithm

Enter the processes in the ready queue in order; P1, P2, P3, P4 and P5.

Each process is given a TQ= 20ms periodically.

Gantt chart

P1	P2	P3	P4	P5	P3	P4	P5	P4	P5

0 15 34 54 74 94 97 117 137 144 160

Figure 24: Gantt chart for RR Algorithm (Case 2)

Average waiting time= 58ms

Average turnaround time= 90ms

b) Adaptive RR algorithm

Arrange the processes in the ready queue in an ascending order of their burst time; P1=15, P2=19, P3=23, P4=47 and P5=56. Since the number of processes is odd, the time quantum is set to be the burst time of the mid process. Hence, TQ=23 according to Adaptive RR algorithm.

Gantt chart

P1	P2	P3	P4	P5	P4	P5	P4	P5

0 15 34 57 80 103 126 149 150 160

Figure 25: Gantt chart for Adaptive RR Algorithm (Case 2)

Average waiting time= 51.2 ms

Average turnaround time= 83.2 ms

c) RRRT algorithm

Arrange the processes in the ready queue according to their given burst time in an ascending order; P1=15, P2=19, P3=23, P4=47 and P2=56. The time quantum is calculated according to the RRRT Algorithm. The TQ is calculated as 16ms.

Gantt chart

P1	P2	P3	P4	P5	P4	P5

0 15 34 57 73 89 120 160

Figure 26: Gantt chart for RRRT Algorithm (Case 2)

Average waiting time= 45.2ms

Average turnaround time= 77.2ms

The following table compares among the three algorithms RR, Adaptive RR and RRRT with respect to average waiting time and average turnaround time.

Table 10. Comparison among the RR, Adaptive RR and RRRT Algorithms

Algorithm	Average waiting time	Average Turnaround time
RR	58	90
Adaptive RR	51.2	83.2
Round Robin Remaining Time	45.2	77.2

Figures 27, 28 draw the comparison of average waiting time and average turnaround time for the RR, Adaptive RR and RRRT algorithms.

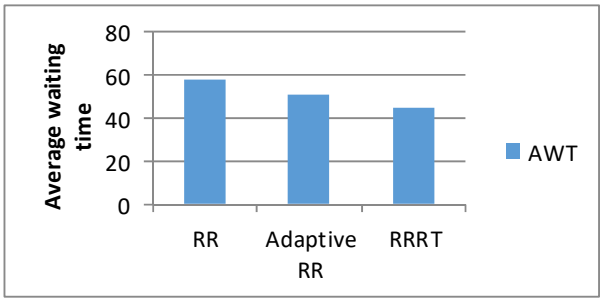

Figure 27: Graph for Average Waiting Time for RR, Adaptive RR and RRRT algorithms (Case 2)

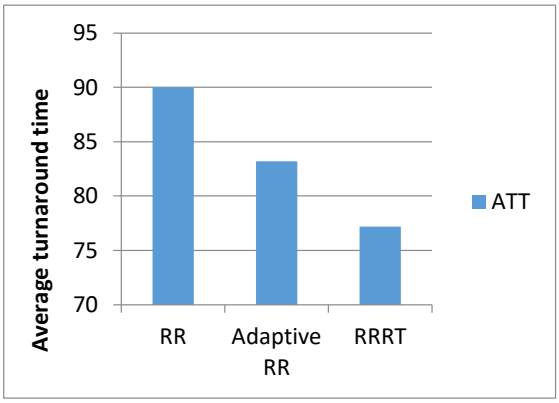

Figure 28: Graph for Average Turnaround Time for RR, Adaptive RR and RRRT algorithms (Case 2)

Case 3: The burst time of the processes is in decreasing order

Consider five processes with a randomly decreasing order of burst time shown in Table 11. We apply the algorithms RR, Adaptive RR and RRRT to schedule these processes and the resulting Gantt charts for the RR, Adaptive RR and RRRT algorithms are shown in Figures 29, 30 and 31 respectively.

Table 11. Burst time in decreasing order

process	Burst time
P1	70
P2	53
P3	26
P4	20
P5	15

a) RR algorithm

Enter the processes in the ready queue in order; P1, P2, P3, P4 and P5.

Each process is given a TQ= 25ms periodically.

Gantt chart

P1	P2	P3	P4	P5	P1	P2	P3	P1	P2

0 25 50 75 95 110 135 160 16 181 184

Figure 29: Gantt chart for RR Algorithm (Case 3)

Average waiting time= 109.4ms

Average turnaround time= 146.2ms

b) Adaptive RR algorithm

Arrange the processes in the ready queue in an ascending order of their burst time; P5=15, P4=20, P3=26, P2=53 and P1=70. Since the number of processes is odd, the time quantum is set to be the burst time of the mid process. Hence, TQ=26 according to Adaptive RR algorithm.

Gantt chart

P5	P4	P3	P2	P1	P2	P1	P2	P1

0 15 35 61 87 113 139 165 166 184

Figure 30: Gantt chart for Adaptive RR Algorithm (Case 3)

Average waiting time= 55.4 ms

Average turnaround time= 92.2 ms

c) RRRT algorithm

Arrange the processes in the ready queue according to their given burst time in an ascending order; P5=15, P4=20, P3=26, P2=53 and P1=70. The time quantum is calculated according to the RRRT Algorithm. The TQ is calculated as 18ms.

Gantt chart

P5	P4	P3	P2	P1	P2	P1

0 15 35 61 79 97 132 184

Figure 31: Gantt chart for RRRT Algorithm (Case 3)

Average waiting time= 48.6ms

Average turnaround time= 85.4ms

The following table compares among the three algorithms RR, Adaptive RR and RRRT with respect to average waiting time and average turnaround time.

Table 12. Comparison among the RR, Adaptive RR and RRRT Algorithms

Algorithm	Average waiting time	Average Turnaround time
RR	109.4	146.2
Adaptive RR	55.4	92.2
Round Robin Remaining Time	48.6	85.4

Figures 32, 33 draw the comparison of average waiting time and average turnaround time for the RR, Adaptive RR and RRRT algorithms.

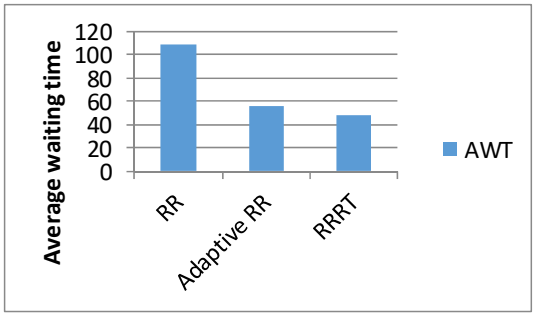

Figure 32: Graph for Average Waiting Time for RR, Adaptive RR and RRRT algorithms (Case 3)

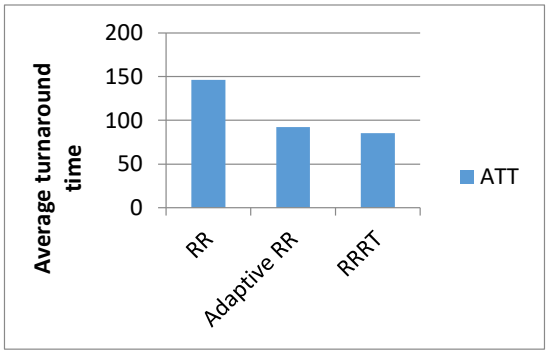

Figure 33: Graph for Average Turnaround Time for RR, Adaptive RR and
RRRT algorithms (Case 3)

3. Dynamic TQ

The performance of RR algorithm depends on the fixed Time Quantum [6]. If it assumes a large TQ, then many processes whose burst time less than TQ will be finished during its TQ. Hence the RR algorithm tends to be a FCFS algorithm. On the other hand if TQ is very small, the processes will be interrupted many times. These interruptions increase the overhead of the CPU which leads to a large number of context switches.

A dynamic TQ may changes many times during the execution of the algorithm depending on the current values of the process's burst time [16].

Many algorithms are developed to improve the efficiency of the RR algorithm using dynamic TQ.

The following is a comparison among the MMRR, IRRVQ, AMRR [17] [18] [19] and RR algorithms.

3.1 Min-Max Dispersion Measure (MMRR)

This algorithm depends on dynamic TQ; the processes are sorted in an increasing order of their burst time. It takes TQ as the range of the CPU burst time of all the processes which is the difference between the maximum and minimum values of the processes' burst time. If this range is less than 25 then the next TQ is set to 25, otherwise the next TQ is set to the calculated TQ [7].

Pseudo code for Min-Max Round Robin (MMRR) algorithm is as follow.

1. All the processes present in the ready queue are sorted in ascending order.
   //n = number of processes, i = loop variable
2. while (RQ != NULL)
   //RQ = Ready Queue
   TQ = MAXBT – MINBT
   //TQ = Time Quantum
   //MAXBT = MAXimum Burst Time
   //MINBT = MINimum Burst Time
   (Remaining burst time of the processes)
   // If one process is there then TQ is equal to BT of itself
3. if (TQ < 25)
   set TQ_{new} = 25
   else
   set TQ_{new} = TQ
   end if
4. //Assign TQ to (1 to n) process
   for i = 1 to n
   {
   $P_i \rightarrow TQ_{new}$
   }
   end for
   // Assign TQ_{new} to all the available processes.
5. Calculate the remaining burst time of the processes.
6. if (new process is arrived and BT != 0)
   go to step 1
   else if (new process is not arrived and BT != 0)
   go to step 2
   else if (new process is arrived and BT == 0)
   go to step 1
   else
   go to step 7
   end if
   end while
7. Calculate ATT, AWT and CS.
   //ATT = Average Turnaround Time
   //AWT = Average Waiting Time
   //CS = number of Context Switches
8. End

Flowchart of Min-Max Round Robin (MMRR) algorithm

3.2 The improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ)

This algorithm associates the features of SJF and RR scheduling algorithms with varying time quantum. First, it arranges all the processes in the ready queue in an ascending order. The processes are allocated to the CPU using RR scheduling with time quantum value equal to the first process's burst time in the ready queue. After each iteration the remaining processes are again arranged in an ascending order and allocated to the CPU using RR scheduling with TQ equal to the burst time of the first process in the ready queue [8].

Pseudo code of the IRRVQ CPU scheduling algorithm is as follow.

1. Make a ready queue RQUEUE of the Processes submitted for execution.
2. DO steps 3 to 9 WHILE queue RQUEUE becomes empty.
3. Arrange the processes in the ready queue REQUEST in the ascending order of their remaining burst time.
4. Set the time quantum value equal to the burst time of first process in the ready queue RQUEUE.
5. Pick the first process from the ready queue RQUEUE and allocate CPU to this process for a time interval of up to 1 time quantum.
6. Remove the currently running process from the ready queue RQUEUE, since it has finished execution and the remaining burst time is zero.

7. REPEAT steps 8 and 9 UNTIL all processes in the ready queue gets the CPU time interval up to 1 time quantum.
8. Pick the next process from the ready queue RQUEUE, and allocate CPU for a time interval of up to 1 time quantum.
9. IF the currently running process has finished execution and the remaining CPU burst time of the currently running process is zero
   remove it from the ready queue
   ELSE
   remove the currently running process from the ready queue RQUEUE and put it at the tail of the ready queue.

3.3 Average Max Round Robin Scheduling Algorithm (AMRR)

This algorithm uses a dynamic TQ to minimize the average waiting time and average turnaround time. All the processes are arranged in an ascending order of their burst time. The TQ is calculated where TQ is equal to (average of burst times + maximum burst time)/2.

AVG = (Summation of Burst Time of all the processes)/ Number of processes
TQ= (AVG+MAXBT)/2

After first iteration the processes are arranged again in an ascending order of their remaining burst time and new TQ is calculated [9].

Pseudo code for Avg Max Round Robin (AMRR) algorithm as follow.

1. All the process in the ready Queue are stored in ascending order
   // n= Number of processes // i= Loop variable (counter) //BT= Burst Time
2. While (RQ!=NULL) // RQ=Ready Queue //TQ=Time Quantum //AVG=Average of all the processes
   //MAXBT=Maximum Burst Time AVG= (Sum of BT of all the process)/Number of processes. //(Use round off function in AVG.) TQ=(AVG+MAXBT)/2
   //(Use round off function in TQ) (Remaining Burst time of the process)
   //If one process is there then TQ is equal to BT itself
3. // Assign TQ to (1 to n) processes
   for i=0 to n loop
   {
   Pi->TQn
   }
   //TQn=New Time Quantum
   end of for
   //Assign TQn to all the available processes
4. Calculate the remaining Burst time of the processes.
5. If (new process arrived and BT!=0 or new process is arrived and BT==0)

   then go to step1
   else if (new process is not arrived and BT!=0) then go to step 2
   else
   go to step 6
   end of if
   end of while
6. Calculate ATT,AWT,CS

   //ATT=Average turnaround time
   //AWT=Average waiting time //CS=Number of context switch
7. End

3.4 Simulation

Consider five processes with random burst time arrived at time 0 as shown in Table 13. The Gantt charts for the RR, MMRR, IRRVQ and AMRR algorithms are shown in Figures 34, 35, 36 and 37 respectively.

Table 13. Burst time for processes

Process	Burst time
P1	20
P2	55
P3	38
P4	24
P5	40

a) RR algorithm

Enter the processes in the ready queue in order; P1, P2, P3, P4 and P5.
Each process is given a TQ= 22ms periodically.

Gantt chart

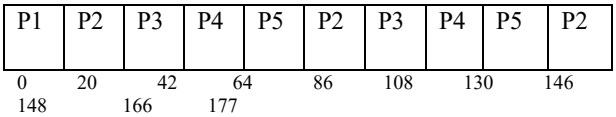

Figure 34: Gantt chart for RR Algorithm

Average waiting time= 109.4ms
Average turnaround time= 146.2ms

b) MMRR algorithm

Arrange all the processes in the ready queue in an ascending order according to their burst time; P1=20, P4=24, P3=38, P5=40 and P2=55. The TQ calculated as the difference between the maximum burst time and the minimum burst time.

Gantt chart

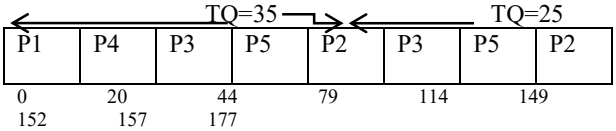

Figure 35: Gantt chart for Adaptive MMRR Algorithm

Average waiting time= 65.8 ms
Average turnaround time= 110 ms

c) IRRVQ algorithm

Arrange all the processes in the ready queue in an ascending order according to their burst time; P1=20, P4=24, P3=38, P5=40 and P2=55. In each iteration, the TQ is equal to the first process's burst time.

Gantt chart

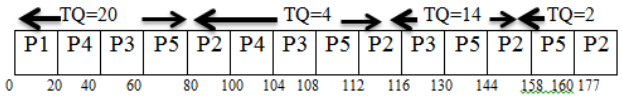

Figure 36: Gantt chart for IRRVQ Algorithm

Average waiting time= 82.8ms
Average turnaround time= 118.2ms

d) AMRR algorithm

Arrange all the processes in the ready queue in an ascending order according to their given burst time in an ascending order; P1=20, P4=24, P3=38, P5=40 and P2=55. The TQ is the mean of (average of BT + Max BT)

Gantt chart

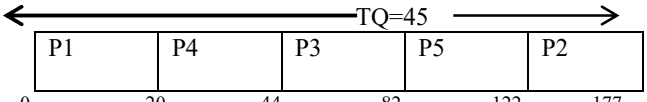

Figure 37: Gantt chart for AMRR Algorithm

Average waiting time= 53.6ms
Average turnaround time= 89ms

Summarizes the results obtained above with respect to average waiting time and average turnaround time.

Table 14. Comparative study of RR, MMRR, IRRVQ and AMRR Algorithms with dynamic TQ

Algorithm	Average waiting time	Average Turnaround time
RR	96	131.4
MMRR	65.8	110
IRRVQ	82.8	118.2
AMRR	53.6	89

Figures 38, 39 below show the comparison of average waiting time and average turnaround time for the RR, MMRR, IRRVQ and AMRR algorithms.

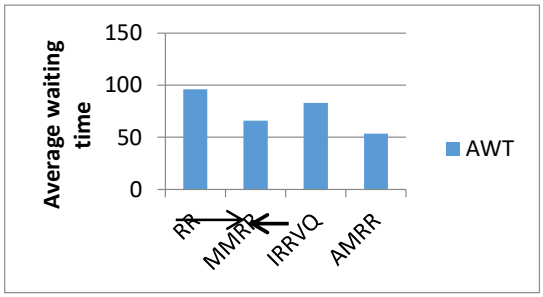

Figure 38: Graph for Average Waiting Time for RR, MMRR, IRRVQ and AMRR algorithms

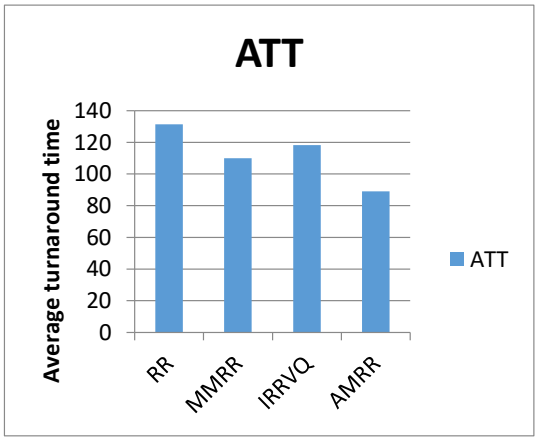

Fig figure 39: Graph for Average Turnaround Time for RR, MMRR, IRRVQ and AMRR algorithms

From the figures 38, 39 it is noticed that the algorithms MMRR, IRRVQ and AMRR are doing better than the standard Round Robin algorithm with respect to the average waiting time and the average turnaround time. By comparing the results of the other three algorithms, it is noticed that the MMRR algorithm reduces the average waiting time and the average turnaround time than the IRRVQ algorithm. However the AMRR algorithm is the best one of them.

4. Results analysis and amelioration

The TQ has two cases; static or dynamic. In the static TQ, each process has the same time slice where TQ may be either randomly chosen or calculated. The dynamic TQ is repeatedly calculated after each iteration.

In the static randomly chosen TQ; by comparing the algorithms RR, IRR, AAIRR and ERR, the AAIRR algorithm is the algorithm which gives the best results in random and decreasing cases. But in the case of increasing order it gives the same results as the ERR algorithm because of the benefit of AAIRR algorithm which is selecting the next short process to assign it to the processor which leads to minimize the average waiting time and average turnaround time does not exist.

In the static calculated TQ; by comparing the algorithms RR, Adaptive RR and RRRT, the algorithm RRRT gives better results than other algorithms. On the other hand in the dynamic TQ algorithms; by comparing the algorithms RR, MMRR, IRRVQ and AMRR, the AMRR algorithm performs better than other algorithms.

The results analysis shows that a combination of calculated the TQ in the static case may be ameliorated through the TQ equation of RRRT algorithm. This will be explained in the next section. Because the calculated TQ is more logic than randomly chosen TQ. So we recommended to use the Enriched Round Robin for minimizing the average waiting time and average turnaround time.

5. Enriched Round Robin algorithm

This algorithm improves the RRRT algorithm. If all the processes arrive at the same time to the ready queue, then all the processes are arranged in an ascending order according to their burst time. The TQ is calculated by (¾*mean). If the remaining CPU burst time of the currently running process is less than the time quantum, the CPU is again allocated to the currently running process for the remaining CPU burst time. Otherwise, the process will be appended to the end of the ready queue. We proposed this algorithm that reduced the average waiting time and the average turnaround time in all cases which is published in [20] and proved that it improved the TQ that achieves stability in terms of the average waiting time and the average turnaround time.

6. Conclusion

Many algorithms are proposed to improve the efficiency of the standard Round Robin algorithm to minimize the average waiting time and average turnaround time. TQ is the most important issue in the performance of these algorithms; it can be static or dynamic. In static TQ, each process has the same time slice where TQ is either can be randomly chosen or calculated. Dynamic TQ is repeatedly calculated after each iteration.

From the above study, in static TQ algorithms it is noted that the algorithm AAIRR reduces the average waiting time and average turnaround time than other algorithms for the randomly chosen TQ. In the calculated TQ, we recommend that the Enriched Round Robin algorithm which gives the best results and achieving stability in terms of the average waiting time and the average turnaround time. On the other hand the dynamic TQ algorithms; the AMRR algorithm performs better than other algorithms.

7. References

[1] Sukumar Babu Bandarupalli, Neelima Priyanka Nutulapati, P.Suresh Varma, "A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive", International Journal of Modern Engineering Research (IJMER), Vol 2, Nov-Dec. 2012.

[2] Rohith Roshan, K. Subra Rao, "Least-Mean Difference Round Robin (LMDRR) CPU Scheduling Algorithm", Journal of Theoretical and Applied Information Technology, Vol 88(1), June 2016.

[3] Mohd Abdul Ahad, "Modifying Round Robin Algorithm for Process Scheduling using Dynamic Quantum Precision", Special Issue of International Journal of Computer Applications (0975 – 888) on Issues and Challenges in Networking, Intelligence and Computing Technologies – ICNICT 2012, November 2012.

[4] Abbas Noon, Ali Kalakech and Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average," International Journal of Computer Science, Vol 8(1), pp. 224-229, May 2011.

[5] Sukumar Babu B., Neelima Priyanka N. and Sunil Kumar B., "Efficient Round Robin CPU Scheduling Algorithm," International Journal of Engineering Research and Development, Vol 4(9), Nov. 2012, p. 36-42.

[6] Christopher McGuire, Jeonghwa Lee, "Comparisons of Improved Round Robin Algorithms", Proceedings of the World Congress on Engineering and Computer Science, Vol I, 2014.

[7] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", (9th Edn. John Wiley and Sons Inc, 2013.

[8] R.N.D.S.S. Kiran, Polinati Vinod Babu, and B.B. Murali Krishna, "Optimizing CPU Scheduling for Real Time Applications Using Mean-Difference Round Robin (MDRR) Algorithm", S.C. Satapathy et al. (eds.), ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of CSI - Volume I, Advances in Intelligent Systems and Computing 248 DOI: 10.1007/978-3-319-03107-1_78, © Springer International Publishing Switzerland 2014.

[9] Anju Muraleedharan, Neenu Antony, R. Nandakumar, "Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time", Indian Journal of Science and Technology, Vol 9(8),pp.1-6, February 2016.

[10] Imran Qureshi, "CPU Scheduling Algorithms: A Survey", Int. J. Advanced Networking and Applications, Vol 5(4), pp.1968-1973 (2014).

[11] Manish Kumar Mishra, Abdul Kadir Khan, "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol 3(6), pp. 64-69, June 2012.

[12] Abdulrazaq Abdulrahim, Salisu Aliyu, Ahmad M Mustapha, Saleh E Abdullahi, "An Additional Improvement in Round Robin (AAAIRR) CPU Scheduling Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 4(2), pp. 64-69, February 2014.

[13] Jayanti Khatri, "An Enhanced Round Robin CPU Scheduling Algorithm", IOSR Journal of Computer Engineering (IOSR-JCE), Vol 18(4), pp. 20-24, Ver. II (Jul.-Aug. 2016).

[14] Saroj Hiranwal, Dr. K.C. Roy "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice", International Journal

of Computer Science and Communication, Vol 2 (2), pp. 319-323, (July- December) 2011.

[15] Arpita Sharma1, Mr. Gaurav Kakhani, "Analysis of Adaptive Round Robin Algorithm and Proposed Round Robin Remaining Time Algorithm", International Journal of Computer Science and Mobile Computing, Vol 4(12), pp.139-147, December 2015.

[16] Rami J. Matarneh,"Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences Vol 6(10), 2009.

[17] Sanjaya Kumar Panda, Sourav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure", International Journal on Computer Science and Engineering (IJCSE), Vol 4(1), pp. 45-53, January 2012.

[18] Manish Kumar Mishra, Dr. Faizur Rashid, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum", International Journal of Computer Science, Engineering and Applications (IJCSEA), Vol 4(4), pp.1-8, August 2014.

[19] Pallab Banerjee, Probal Banerjee, Shweta Sonali Dhal, "Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol 1(3), pp. 56-62, August 2012.

[20] Kamal ElDahshan, Afaf Abd El-kader, Nermeen Ghazy, "Achieving Stability in the Round Robin Algorithm", International Journal of Computer Applications, Vol 172 (6), pp.15-20, August 2017.

[21] Nermeen Ghazy, 2017 Enhancing Scheduling Jobs on Different Structures of Operating System. Master thesis. University of Al-Azhar.

Biographies

Prof. Kamal Abdelraouf ElDahshan

He is a professor of Computer Science and Information Systems at Al - Azhar University in Cairo, Egypt. An Egyptian national and graduate of Cairo University, he obtained his doctoral degree from the Universitè de Technologie de Compiègne in France, where he also taught for several years. During his extended stay in France, he also worked at the prestigious Institute National de Tèlècommunications in Paris. Professor ElDahshanâAZs extensive international research, teaching, and consulting experiences have spanned four continents and include academic institutions as well as government and private organizations. He taught at Virginia Tech as a visiting professor; he was a Consultant to the Egyptian Cabinet Information and Decision Support Centre (IDSC); and he was a senior advisor to the Ministry of Education and Deputy Director of the National Technology Development Centre. Professor ElDahshan is a professional Fellow on Open Educational Resources as recognized by the United States Department of State and an Expert at ALECSO as recognized by the League of Arab States.

Dr. Afaf Abd El-Kader Abd EL-Hafiz

Is currently lecturer at University of Al-Azhar. She is doing her research work in Scheduling algorithms.

Nermeen Alaa Eldin Ghazy

She received her B.Sc. from Faculty of Science, AL-Azhar University at 2011. and Ms. Ghazy received M.Sc in scheduling algorithms at 2017.